Extensible Markup Language (XML)

A promessa e a esperança?…

José Carlos Leite Ramalho – jcr@di.uminho.pt

Departamento de Informática
Escola de Engenharia
Universidade do Minho
Portugal

Introdução

Hoje, a Internet é um enorme mercado em crescimento constante que continuará a crescer no futuro à medida que mais pessoas se forem ligando. Este mercado global tornou-se atractivo para as mais variadas empresas que provocaram a criação de cada vez mais páginas de informação para o "World Wide Web" (WWW). Esta explosão de interesse e investimento levou a um desenvolvimento dos standards e estruturas associados ao WWW. A cada dia que passa novos standards são criados com a promessa de que irão facilitar a vida daqueles que têm de produzir páginas para o WWW. Mas será que estes standards cumprem o que prometem?

Durante muitos anos, a única linguagem à disposição do público, para criação de páginas, foi a "HyperText Markup Language" (HTML). Hoje, temos um conjunto enorme e variado de standards: "Dynamic HTML" (DHTML), "Common Object Request Broker Architecture" (CORBA), "Distributed Component Object Model" (DCOM), "WebGraphics", CGM, SVG, para mencionar alguns. O que problema que se coloca a quem vai desenvolver um "website" é qual ou quais escolher: quais as limitações, possibilidades, vantagens e desvantagens das diferentes tecnologias. Isto implica um conhecimento de todas as arquitecturas para escolher a que melhor se adapta ao problema em causa.

No meio desta confusão surge mais um standard: "eXtensible Markup Language" (XML). Desenvolvido pelo "World Wide Web Consortium" (W3C), surge com a promessa de se tornar o standard da Internet e de todas as transferências de dados estruturados.

Este curso incidirá sobre a temática do XML e terá as seguintes partes:

- O que é o XML? - sintaxe e exemplos.
- Qual a sua relação com outros standards, nomeadamente: XSL, XQL, Xpointer.
- Que utilizações poderemos dar a este standard.
- Apresentação de algumas aplicações que suportam o ciclo de vida dum documento XML: editores, motores de transformação, browsers, …

No fim, pretende-se que a audiência fique com uma ideia concreta do que é o XML, onde deve ser aplicado, e que passos terão de ser seguidos para utilizar esta tecnologia.

XML: o que é?

Origem

A linguagem de anotação XML foi definida pelo W3C [http://www.w3c.org], que já antes havia definido o HTML, a linguagem mais utilizada até hoje na construção de páginas para a Internet.

World Wide Web Consortium (W3C)

O W3C tem como membros 275 organizações que incluem empresas, organizações não lucrativas, grupos industriais e agências governamentais de todo o mundo.
Para um grupo que reune tanta gente de tão grande importância, o W3C é pouco conhecido. As suas reuniões são fechadas a apenas membros do consórcio, o que tem levantado muitas críticas por parte da opinião pública.
O W3C não é uma organização de emissão de stantards como o "American National Standards Institute" (ANSI) ou o "International Standards Organization" (ISO). Pode ser visto apenas como um grupo de tecnólogos que emite recomendações para a criação de novos standards relacionados com a Internet.

O Passado

O XML não é mais do que um subconjunto de outro standard lançado em 1986, o "Standard Generalized Markup Language" (SGML). O SGML foi o primeiro standard de linguagens de anotação de texto e o seu objectivo era o de normalizar a produção de documentos. Para isso dispunha e dispões de uma série de características invulgares das quais destacamos:

- Separação completa do conteúdo e da aparência visual – o SGML só se preocupa com a estrutura da informação e não com a forma em que esta vai aparecer ao utilizador.
- É completamenete independente de plataformas de hardware e software – um documento SGML pode ser transportado entre sistemas ou aplicações sem a alteração duma vírgula.

O SGML e o XML são mais do que linguagens de anotação, são meta-linguagens; permitem a definição de novas linguagens.
Por outro lado, o HTML, a linguagem de anotação mais conhecida para a produção de páginas para a Internet, está definida em SGML. É uma linguagem concreta, não é possível extendê-la a não ser que se altere a sua definição inicial (o que a transformaria noutra linguagem).
O XML não tem esta limitação. Sendo uma meta-linguagem, permite em qualquer momento o acrescentar de novos elementos à linguagem. Na prática, isto traduz-se numa linguagem aberta que nos permite especificar qualquer coisa com o nível de detalhe que pretendermos.
Neste momento, o HTML continua a ser a linguagem mais utilizada na Internet mas, irá ser gradualmente substituído pelo XML.

Porquê XML?

O crescimento espantoso da Internet, nomeadamente do serviço WWW, nos últimos anos, é um facto que se deve principalmente à possibilidade de distribuir facilmente e a baixos custos informação e aplicações, a qualquer utilizador em qualquer parte do mundo.
À medida que a informação que se coloca na Internet se vai tornando cada vez mais complexa, e o número de utilizadores vai crescendo, as limitações das tecnologias e standards actuais vão-se tornando cada vez mais evidentes.

A limitação na construção de páginas WWW deve-se principalmente ao facto do HTML possuir apenas um conjunto fixo e pré-definido de etiquetas com o qual se pode definir a estrutura e a aparência duma página WWW. Foi esta limitação, a impossibilidade de de criar extensões à linguagem que tornou a criação dum novo standard desejável. Nalgumas aplicações, principalmente onde uma publicação electrónica de alta qualidade e desempenho é o objectivo a atingir, tem-se vindo a adoptar o SGML para a produção e manutenção de conteúdos; as páginas WWW são depois geradas automaticamente, partindo destes conteúdos. No entanto, o SGML é complexo e de difícil aprendizagem o que fará com que seja apenas utilizado por comunidades que possuam um elevado nível técnico, e portanto, não tenha uma grande aceitação por parte dos utilizadores normais.

Nos últimos tempos, as aplicações distribuídas em geral e o Java em particular, vieram realçar a falta dum standard para a transferência de dados estruturados. Apesar dos esforços e do dinheiro investido esse standard não surgiu. Os problemas surgem quando diferentes aplicções querem interagir; normalmente, cada uma tem o seu próprio formato para comunicar dados que é diferente do da outra.

Para resolver estes problemas, foi criado o XML. O XML oferece um método estruturado e consistente para descrever e transferir informação. A melhor característica que possui, herdada do SGML, é a separação do formato visual da informação propriamente dita. Isto faz com que o XML seja a linguagem ideal para a produção de conteúdos textuais (inependência de plataformas de hardware e software, longevidade, …). Duas aplicações XML podem enviar e receber informação livremente sem preocupações com o formato dessa informação, a informação contida num documento XML auto-descreve-se.

O XML foi concebido tendo uma série de objectivos em vista:

- Deve ser directamente utilisável na Internet – os utilisadores devem ser capazes de ver páginas XML da mesma maneira que vêem páginas HTML.
- Deve suportar uma série de aplicações: editores, browsers, sistemas de gestão de bases de dados, … No entanto, o principal objectivo é a produção de conteúdos estruturados para a Internet.
- Deve ser compatível com o SGML – já existem muitos conteúdos em SGML e para quem os produziu é a compatibilidade entre os dois é crítica.
- A escrita de programas para processar documentos XML deve ser simples.
- O número de catacterísticas opcionais deve ser reduzido a um mínimo, pois quantas mais houver mais problemas de compatibilidade poderão surgir.

XML: características básicas

O XML foi inicialmente definido como uma sintaxe universal para representar informação estruturada. A estruturação dum documento é complexa e pode ser vista sob três paradigmas diferentes: estrutura baseada no conteúdo, no significado, ou na utilização que se quer dar à informação. Assim, um documento XML é composto por duas coisas: o conteúdo propriamente dito e as anotações/etiquetas que marcam a estrutura.

A maneira mais simples de descrever o XML é compará-lo com o HTML (que quase toda a gente conhece). À semelhança do HTML, a estrutura dum documento XML é construída recorrendo a anotações. As anotações em XML não têm nenhum significado pré-definido. Além disso, o XML é extensível, estruturado, e pode ser validado. Como já foi dito, o XML separa o que é conteúdo do que é material de apresentação gráfica, o que faz com que o mesmo documento possa ser apresentado de várias formas diferentes sem nenhuma alteração do conteúdo estruturado.

Os dois exemplos seguintes mostram as diferenças entre o HTML e o XML.

Exemplo: Sumário de aula (HTML)

```
  <H1>Escola de Verão</H1>
  <H2>Aula 1 – 23.11.1999</H2>
  <H2>Prof: José Carlos</H2>
  <H2>Tipo: Teórico-Prática</H2>
  <P> Introdução ao XML.
  <P> Conceito de DTD e de documento estruturado.
```

Exemplo: Sumário de aula (XML)

```
  <Curso>
    <Nome> Escola de Verão</Nome>
    <Aula>
      <Nome> Aula 1 – 23.11.1999</Nome>
      <Professor>José Carlos</Professor>
      <Tipo>Teórico-Prática</Tipo>
      <Sumário>
        <para>Introdução ao XML.</para>
        <para> Conceito de DTD e de documento
estruturado.</para>
      </Sumário>
    </Aula>
    <Aula> …
    </Aula>
  </Curso>
```

Como se pode ver nenhuma das anotações tem informação sobre aparência, apenas contêm informação sobre a estrutura e conteúdo do documento. A formatação é deixada ao critério da aplicação que irá utilizar o documento. O nome de cada anotação pode ser escolhido

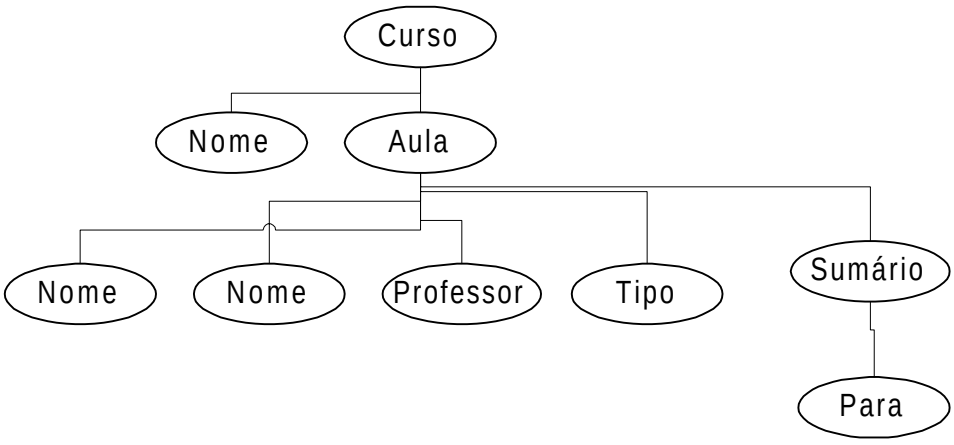
livremente pelo criador do documento apenas deverá obedecer ao princípio de ser descritivo em relação ao conteúdo do elemento respectivo.

Elementos

Como foi mostrado nos exemplos anteriores, a estrutura dum documento XML é construída à custa de anotações que se colocam no texto e que agrupam frases e palavras. É a estes grupos que se dá o nome de elementos. Cada elemento representa um componente lógico do documento. Um elemento pode ter uma das seguintes formas:

- Uma anotação e respectivo texto:
  ```
  <Tipo>Teórico-Prática</Tipo>
  ```
- Uma anotação que contém outros elementos (que por sua vez podem conter outros elementos e assim recursivamente):
  ```
  <Sumário>
    <para>Introdução ao XML.</para>
    <para> Conceito de DTD e de documento
  estruturado.</para>
  </Sumário>
  ```

Podemos assim concluir que um documento XML tem uma estrutura lógica em forma de árvore. Há sempre um elemento que contem todos os outros (o elemento raíz correspondente à raíz da árvore estrutural). A estrutura em árvore do documento é utilizada



pela maior parte das aplicações para ler ou processar o documento.
A título de exemplo, apresenta-se na figura acima a estrutura do documento usado nos exemplos anteriores.

Regra geral, os valores guardados em cada um dos elementos podem ser de três tipos diferentes: texto normal, texto monobloco (as ferramentas deverão tratá-lo como um bloco sem tentar processar o seu conteúdo), e instruções de processamento:

- PCDATA – texto normal.
- CDATA – texto monobloco; iniciado por `<![CDATA[` e terminado por `]]>`.

- PI – instruções de processamento; servem para passar informção às ferramentas que irão processar o documento; são iniciadas por <? e terminadas por ?>; um exemplo muito conhecido é a instrução de processamento que inicia todos os documentos XML, e que indica ao parser qual a versão XML a que o documento obedece:
  ```
  <?XML version=”1.0”?>
  ```

Atributos

Para além do conteúdo, os elementos podem ter outro tipo de informação a eles associada. Esta informação recebe, normalmente, a designação de atributos. Os atributos servem para descrever propriedades dos elementos. Muitas vezes, surge a discussão se um determinado item de informação poderá ser um subelemento ou um atributo do elemento corrente (a decisão irá depender apenas da subjectividade do criador).

Em XML os atributos têm uma sintaxe semelhante à utilizada para os atributos do HTML. Apresenta-se a seguir um exemplo.

Exemplo: O ficheiro de alunos

```
<Alunos>
  <!– elemento aluno tem 2 atributos:
        número e email →
  <Aluno numero=”4140”
         email=jcr@di.uminho.pt>
    <Nome>José Carlos Ramlho</Nome>
  </Aluno>
  <Aluno numero=”15640”
         email=garnold@zaz.br>
    <Nome>Gustavo Arnold</Nome>
  </Aluno>
</Alunos>
```

Entidades

Um documento XML pode ser definido como uma série linear de caracteres e referências a outros objectos. Um processador que vá processar documentos XML começa pelo início e vai seguindo as referências até ter terminado de processar todo o documento. As entidades compõem um mecanismo de organização dos objectos dum documento não-linear. Caberá depois ao parser reorganizar o conteúdo numa forma linear. Uma entidade pode ser tão pequena como um caracter ou tão grande como um documento XML. Na terminologia do XML, um ficheiro pode ser partido em vários bocados que serão depois armazenados no disco do computador ou numa base de dados, cada um desses bocados é designado por entidade. As entidades podem ainda estar espalhadas por vários sites da Internet.

Simplificando, uma entidade é composta por um nome e um conteúdo. O conteúdo é a informação propriamente dita, e o nome, é utilizado para referenciar essa informação.

Há vários tipos de entidade, que se podem utilizar em diferentes situações.

Se na declaração duma entidade também definirmos o seu conteúdo, a entidade é designada por entidade interna.

Exemplo: Entidades Internas

- Utilizadas como abreviaturas
  `<!ENTITY xml "eXtensible Markup Language">`
  Sempre que o parser encontrar a referência "&xml;" irá expandi-la para o texto "eXtensible Markup Language".
- Abreviaturas para texto com anotações
  `<!ENTITY xsite "<URL DEST=www.xml.com>XML website</URL>`

Quando o conteúdo duma entidade provém dum ficheiro externo, aquela diz-se externa. A localização do conteúdo é dada um identificador externo: a palavra reservada SYSTEM seguida por um "Uniform Resource Identifier (URI).

Exemplo: Entidades Externas

- Com conteúdo XML
  `<!ENTITY capítulo SYSTEM "cap1.xml">`
  O conteúdo da entidade é o ficheiro XML cap1.xml
- Com conteúdo não XML: imagens, …
  `<!ENTITY foto SYSTEM "foto.gif" NDATA GIF>`
  A entidade corresponde a uma imagem GIF; a palavra reservada NDATA indica que o conteúdo da entidade não é textual e é seguida do identificador de tipo, neste caso GIF.

Uma entidade pode ter várias utilidades:

- Podemos partir um livro nos seus capítulos e geri-los individualmente.
- Podemos criar entidades para os acrónimos mais utilizados; desta maneira garantimos que eles são sempre escritos da mesma maneira e, mais tarde, se pretendermos alterar um deles, basta alterar a definição da entidade e todas as ocorrências serão actualizadas.
- Podemos criar entidades para blocos de texto que se repetem em vários documentos, por exemplo, uma nota de copyright; desta maneira estaríamos a reutilizar.
- Podemos criar entidades para integrar imagens e objectos multimédia nos nossos documentos.
- E outras que a imaginação ditar …

Definição de Tipo de Documento (DTD)

Uma parte importante que um documento XML pode ter, é, à cabeça do documento, a definição da estrutura desse documento. Esta informação, é uma herança do SGML e recebe a designação de Definição de Tipo de Documento ou, em inglês, "Document Type Definition".

O DTD define quais as anotações que podem ser utilizadas no documento, em que ordem podem aparecer e, qual a sua estrutura hierárquica. Com um DTD, o parser pode verificar se o conteúdo do documento está escrito de acordo com esse DTD.

O exemplo seguinte apresenta o DTD para o caso das aulas.

```
Exemplo: DTD para a descrição de aulas
  <!DOCTYPE Curso [
    <!ELEMENT Curso (Nome,Aula+)>
    <!ELEMENT Nome  (#PCDATA)>
    <!ELEMENT Aula
               (Nome,Professor,Tipo,Sumário)>
    <!ELEMENT Professor (#PCDATA)>
    <!ELEMENT Tipo      (#PCDATA)>
    <!ELEMENT Sumário   (para+)>
    <!ELEMENT para      (#PCDATA)>
```

O DTD é usado para definir a estrutura a que a restante informação do documento deve obedecer. O DTD pode ser externo (estar armazenado num ficheiro à parte) ou residir internamente no documento. Hoje em dia, assiste-se a um grande esforço no desenvolvimento de DTDs standard para as mais variadas áreas. Como exemplos práticos temos: o "Text Encoding Initiative (TEI)" para as ciências humanas, o EAD para informação de arquivos, o DocBook para a produção de livros técnicos, o H7 para os registos clínicos, o ATA para as companhias aéreas, …

Válido versus Bem-Estruturado

Numa linguagem, qualquer que ela seja, precisamos de ter uma entidade que verifique se um dado documento escrito nessa linguagem está de acordo com a gramática e especificações dessa linguagem. No nosso caso, o DTD corresponde à gramática e o parser XML o verificador.

No entanto, o XML foi pensado para suportar aplicações "Web" e, em muitos casos, o DTD e as respectivas verificações podem tornar-se um fardo bem pesado. Então criaram-se dois patamares de validação: o documento válido e o documento bem-formado. Um documento diz-se válido se tiver um DTD associado e se o texto do documento está de acordo com as especificações no DTD; um documento diz-se bem-formado se obedecer às seguintes condições:

- Tiver um ou mais elementos.
- Há apenas um elemento (o elemento raíz) para o qual nem a anotação de início nem a anotação de fim estão dentro de qualquer outro elemento.
- Todas as outras anotações têm que estar aninhadas correctamente.
- Todas as entidades usadas no documento têm de estar definidas em XML ou no DTD.

Podemos ver agora dois exemplos: o primeiro dum documento bem estruturado e o segundo do mesmo documento mas agora com o DTD associado ilustrando o conceito de documento válido.

```
Exemplo: Um documento bem estruturado
<RECEITAS>
  <TITULO> O Meu Livro de Receitas </TITULO>
  <RECEITA ORIGEM="Portugal">
    <TITULO> Bolo </TITULO>
    <INGREDIENTE> 500g de farinha </INGREDIENTE>
    <INGREDIENTE> 200g de açucar </INGREDIENTE>
    <INGREDIENTE> 300g de manteiga </INGREDIENTE>
  </RECEITA>
</RECEITAS>

Exemplo: Um documento válido
<!DOCTYPE RECEITAS [
<!ELEMENT RECEITAS    (TITULO,RECEITA*)>
<!ELEMENT TITULO      (#PCDATA)>
<!ELEMENT RECEITA     (TITULO,INGREDIENTE*)>
<!ELEMENT INGREDIENTE (#PCDATA)>
<!ATTLIST RECEITA ORIGEM CDATA #IMPLIED>
]>
<RECEITAS>
  <TITULO> O Meu Livro de Receitas </TITULO>
  <RECEITA ORIGEM="Portugal">
    <TITULO> Bolo </TITULO>
    <INGREDIENTE> 500g de farinha </INGREDIENTE>
    <INGREDIENTE> 200g de açucar </INGREDIENTE>
    <INGREDIENTE> 300g de manteiga </INGREDIENTE>
  </RECEITA>
</RECEITAS>
```

DTDs e Modularidade

O XML permite a combinação de vários DTDs no mesmo documento através da utilização de entidades. Isto é importante, especialmente em grandes projectos pois permite o desenvolvimento modular. O futuro irá com certeza convergir num repositório de pequenos DTDs ou componentes, os quais se poderão combinar para criar novos e maiores componentes.

```
Exemplo: Combinação de vários DTDs
<!DOCTYPE Dicionário [
<!ENTITY abertura SYSTEM “abertura.dtd”>
<!ENTITY corpo    SYSTEM “corpo.dtd”>
<!ENTITY fecho    SYSTEM “fecho.dtd”>
```

```
<!ELEMENT Dicionário (%abertura, %corpo, %fecho)>
]>
```

XML e Tipos de Dados

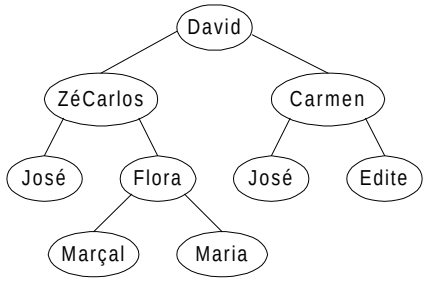
Sempre que um informático é confrontado com uma nova linguagem, a primeira coisa que faz é tentar ver quais os tipos de dados suportados na linguagem (inteiros, strings, listas, arrays, etc). Se tentássemos fazer o mesmo relativamente ao XML depressa constataríamos que o XML suporta todos os tipos de dados e nenhum. A associação de um tipo de dados a um elemento é da responsabilidade da aplicação que irá processar o documento. O parser XML não faz qualquer validação de tipos de dados. A única validação que é feita é estrutural (documento válido ou bem estruturado).

No entanto, há uma série de aplicações onde é preciso validar o conteúdo de alguns elementos. Nessas aplicações, os tipos de dados tornam-se vitais pois para exprimir uma restrição sobre o conteúdo de um elemento, preciso de ter um tipo de dados associado a esse elemento. Com a diversidade de aplicações do XML, esta necessiade cresceu ao ponto de existir já uma proposta de standard para a adição de tipos de dados a documentos XML: XML-Data.

Tipos de Dados Estruturados

Vamos, por um momento, esquecer os tipos atómicos de dados (inteiros, strings, caracteres, reais), e olhar apenas para os tipos de dados estruturados. Como o próprio nome indica têm uma estrutura subjacente e o XML é uma linguagem para descrever estruturas. Então, deverá ser possível descrever um tipo de dados estruturado em XML.
A seguir, apresenta-se um exemplo dum documento XML que descreve uma árvore genealógica.

Exemplo: Uma árvore binária em XML



A figura mostra uma representação gráfica da estrutura que se vai agora definir e instanciar em XML:

```
<?xml version=”1.0”?>
<!DOCTYPE arv-bin [
<!ELEMENT arv-bin (nodo)>
<!ELEMENT nodo    (valor,filhos?)>
```

```
<!ELEMENT valor   (#PCDATA)>
<!ELEMENT filhos  (nodo?,nodo?)>
]>

<arv-bin>
  <nodo>
    <valor>David</valor>
    <filhos>
      <nodo>
        <valor>ZéCarlos</valor>
        <filhos>
          <nodo>
            <valor>José</valor>
          </nodo>
          <nodo>
            <valor>Flora</valor>
            <filhos>
              <nodo>
                <valor>Marçal</valor>
              </nodo>
              <nodo>
                <valor>Maria</valor>
              </nodo>
            </filhos>
          </nodo>
        </filhos>
      </nodo>
      <nodo>
        <valor>Carmen</valor>
        <filhos>
          <nodo>
            <valor>José</valor>
          </nodo>
          <nodo>
            <valor>Edite</valor>
          </nodo>
        </filhos>
      </nodo>
    </filhos>
  </nodo>
```

Este exemplo conclui a introdução básica ao XML. No fim deste documento, será colocada uma lista de bibliografia aconselhada a quem quiser aprofundar mais os conhecimentos deste assunto.

Especificações relacionadas com o XML

As especificações que se descrevem estão ainda em diferentes estágios de desenvolvimento, podem por isso, ser alteradas. Dependendo do grau de desenvolvimento uma especificação está colocada num determinado nível de desenvolvimento. Isto não implica que não estejam a ser utilizadas, muito pelo contrário, há especificações ainda no primeiro nível de desenvolvimento já implementadas.

Apresenta-se a seguir uma lista dos níveis de desenvolvimento e correspondentes especificações. Estas são depois explicadas uma a uma.

Actualmente, o W3C considera quatro níveis de desenvolvimento:

- Recomendação – significa que a especificação está estável, contribui para uma evolução da tecnologia e é suportada pelos membros do W3C que apoiam a sua aplicação.
  - → Extensible Markup Language (XML) 1.0
  - → Document Object Model (DOM) Level 1
  - → Cascading Style Sheets Level 1 (CSS1)
  - → Cascading Style Sheets Level 2 (CSS2)
  - → NameSpaces in XML (XML Namespace)
- Recomendação Proposta – significa que a especificação está a ser revista pelos membros do W3C.
  - → Neste momento não há nenhuma relacionada com o XML.
- Rascunho – significa que a especificação pode ser alterada em qualquer altura, substituída ou simplesmente posta de lado; este tipo de material não deve ser usado como referência podendo, no entanto, ser citado como "trabalho em desenvolvimento".
  - → Extensible Stylesheet Language (XSL)
  - → XML Linking Language (XLL)
  - → XML Pointer Language (XPointer)
- Notas – significa que se trata de uma especificação submetida ao W3C pelo público ou por algum membro; não foi submetida num "Call for proposals"; por ter interesse e qualidade é publicada e mantida para discussão.
  - → XML Query Language (XQL)
  - → XML Data (XML-Data)
  - → Schema for Object-oriented XML (SOX)
  - → Vector Markup Language (VML)

Document Object Model (DOM)

Esta especificação define uma interface e uma plataforma neutras que irão permitir que programas e scripts acedam e alterem o conteúdo, a estrutura e o estilo dos documentos, duma forma normalizada. Isto permitirá uma maior portabilidade de programas e scripts.

Apesar do seu nome, o DOM não é um modelo orientado a objectos e sim, uma maneira independente de declarar interfaces e objectos para manipulação de documentos XML e HTML.

Vão ser disponibilizados três métodos para utilizar o DOM para aceder a documentos XML:

- Utilizando Javascript ou Vbscript numa página Web.
- Utilizando uma aplicação tipo "plug-in", ou ActiveX, que acedem ao documento através dum browser.
- Utilizando um parser XML externo que implementa o DOM.

Quer o Netscape Navigator, quer o Internet Explorer, suportam o DOM, mas com várias alterações não normalizadas. No entanto, ficou estabelecido que nas próximas versões dum e doutro, o DOM suportado seria o standard.

Linguagens para Especificação de Estilo

Como já foi afirmado anteriormente, uma das características mais importantes do XML é a separação entre o que é informação/conteúdo e forma/aparência visual. O que faz com que cada utilizador possa definir o modo como quer ver a informação. Surge então a pergunta: se os documentos XML não têm qualquer informação sobre a sua formatação, de onde vem essa informação?

Essa informação vem duma especificação de estilo que é guardada num ficheiro à parte. Uma especificação de estilo permite definir a apresentação dum documento; é composta por um conjunto de regras que descrevem como cada um dos elementos num documento XML devem ser compostos graficamente. É possível especificar propriedades gráficas como: o tamanho da letra, a cor, o espaçamento do texto, a colocação de imagens, etc.

Neste momento, há duas linguagens de especificação de estilo para documentos XML em desenvolvimento: Cascading Style Sheets (CSS) e Extensible Stylesheet Language (XSL). Ambas serão descritas em mais detalhe nas próximas secções.

Neste momento, uma questão poderia ser levantada: porquê desenvolver duas linguagens e não apenas uma? O quadro seguinte fornece a resposta.

	CSS	**XSL**
Pode ser usada com HTML?	sim	não
Pode ser usada com XML?	sim	sim
Pode transformar documentos?	não	sim
Sintaxe	CSS	XML

Da análise do quadro, pode-se concluir que o CSS pode ser usado para formatar documentos XML e HTML, no entanto, não pode ser usado para transformar documentos. Por outro lado, o XSL apenas pode ser usado para formatar XML e pode transformar documentos. Podemos agora perguntar: quando usar CSS e quando usar XSL?

Sempre que o documento final fôr uma versão estilizada do documento original, devemos usar o CSS. Quando o documento final resultar duma transformação do documento original (construção de índices, listas de nomes, ...), deve ser usado o XSL.

Cascading Style Sheets (CSS)

O CSS é uma linguagem declarativa muito simples que permite, a autores e utilizadores, associar informação de estilo a documentos estruturados XML ou HTML.

O estilo de um elemento especifica-se associando-lhe propriedades e valores.

```
Exemplo: Especificação de estilo para um elemento
  H1 {
       font-size: 16pt;
       font-weight: bold;
       color: blue;
     }
```

Neste exemplo, está-se a declarar que os elementos H1 devem ter o texto em 16 pontos, letra carregada e de côr azul.
As outras propriedades do CSS permitem especificar tudo desde o tamanho de letra dum parágrafo até margens, espaçamento de linhas e texturas de fundo.

O CSS tem dois níveis de especificação suportados em duas implementações diferentes:

- CSS Level 1 (CSS1) – O CSS1 é uma linguagem de fácil leitura e permite expressar o estilo numa terminologia comum a sistemas normais de publicação. Como o nome indica, várias especificações de estilo podem ser aninhadas para produzir o resultado final.
- CSS Level 2 (CSS2) – O CSS2 define um nível de abstracção acima do CSS1 – dá um maior controle ao utilizador na disposição dos objectos na página. Com poucas excepções, todas as especificações CSS1 são especificações CSS2. Na secção seguinte descreve-se melhor o CSS2.

Formatar documentos XML com CSS2

O CSS pode ser usado com qualquer documento estruturado. No nosso caso interessa-nos ver a sua aplicação ao XML.

```
Exemplo: Documento XML
  <?xml version=”1.0”?>
  <CURSO>
    <TITULO>XML, uma promessa?…</TITULO>
    <AUTOR>José Carlos Ramalho</AUTOR>
    <RESUMO>
      <PARA>Durante este curso, faremos uma travessia do
universo <ACRON>XML</ACRON> … </PARA>
```

```
    …
  </RESUMO>
  …
</CURSO>
```

Para formatar este documento recorrendo ao CSS2 temos de fazer duas coisas:

- Criar uma especificação de estilo
- Associar essa especificação de estilo ao documento XML

Especificação de Estilo

A especificação de estilo é simplesmente um ficheiro de texto com extensão ".css". Para este exemplo criou-se o ficheiro "curso.css".

```
Exemplo: curso.css
  ACRON {display: inline}
  CURSO, TITULO, AUTOR, RESUMO, PARA {display: block}
  TITULO {font-size: 16pt}
  AUTOR {font-style: italic}
  CURSO, TITULO, AUTOR, RESUMO, PARA {margin: 2cm}
```

As primeiras duas linhas especificam se os elementos são "inline" (devem ser colocados na mesma linha que os caracteres que os precedem e antecedem), ou "block" (há uma quebra de linha no início do elemento e outra no fim).
As outras linhas alteram o valor de algumas propriedades dos elementos a que se referem (tamanho de letra, postura, margens).

Associar uma Especificação de Estilo

Neste momento, a maneira de associar uma especificação de estilo a um documento XML é inserir no início do documento uma instrução de processamento com um determinado formato:

```
Exemplo: Associar uma especificação de estilo
  <?xml version=”1.0”?>
  <?xml:stylesheet type=”text/css” href=”curso.css”?>
  … Resto do documento …
```

Extensible Stylesheet Language (XSL)

O CSS é uma boa opção para a especificação do estilo sempre que os elementos (parágrafos, listas, cabeçalhos, imagens, tabelas, …) sejam formatados e apareçam no documento resultado na mesma ordem em que se encontram no documento original.
Mas, nem sempre é assim. Há situações em que se pretende reordenar estruturalmente o conteúdo dum documento. Pode-se, por exemplo, querer gerar uma síntese do documento

original, ou uma tabela de referências. Este tipo de operações requer uma transformação do documento original. O XSL permite especificá-las.

O XSL adoptou a sintaxe do XML. No entanto, encontra-se ainda em fase de desenvolvimento, podendo ser alterado a qualquer momento. Este facto não assustou, por exemplo a Microsoft que já integrou no seu browser ("Internet Explorer") a possibilidade de processar estilos especificados em XSL.

O XSL resultou da fusão de algumas características de duas outras linguagens de especificação de estilo: do DSSSL – "Document Style Semantics Specification Language", e do já descrito CSS.

O DSSSL é um standard ISO que foi criado para normalizar toda e qualquer transformação e formatação de documentos estruturados. É composto por quatro sub-linguagens: uma linguagem para especificação de transformações, uma linguagem para especificação de estilo, uma linguagem de query, e uma linguagem funcional que oferece um motor de cálculo e liga as outras, o Scheme. Tudo isto faz do DSSSL uma linguagem poderosa e muito complexa. Principalmente devido a este factor, complexidade, a sua divulgação e aceitação por parte dos autores e utilizadores tem sido lenta. Para uma abordagem mais profunda da linguagem aconselhámos a leitura das referências incluídas no fim em Bibliografia.

O XSL combina parte das potencialidades do DSSSL com o CSS e adiciona uma linguagem de programação para ligá-las, o ECMAScript, uma versão standard de Javascript.

Para construir um determinado output da informação guardada num documento XML, um processador XSL irá utilizar uma especificação de estilo para fazer uma travessia ao documento e produzir o output desejado.
Até ao momento, os processadores XSL disponíveis geram apenas outputs em HTML, mas em teoria poderiam gerar qualquer outro formato (à semelhança do DSSSL): RTF, ASCII, PDF, TEX, etc.
Num futuro muito próximo, o XSL (na versão corrente ou com alterações) será suportado directamente nos browsers, de momento isso não acontece.

Em XSL, a associação de uma folha de estilo a um documento XML é feita da mesma maneira que em CSS, através da inclusão da linha:

Exemplo: Associação de uma folha de estilo

```
<?xml-stylesheet type="text/xsl" href="curso.xsl"?>
```

Anatomia de uma folha de estilo XSL

Eis o conceito mais importante subjacente ao XSL: o XSL não manipula elementos, manipula objectos gráficos ("flow objects"). Mais especificamente, transforma bocados dum documento XML em objectos gráficos.

Surge então a questão: O que são de facto estes objectos gráficos?

Um objecto gráfico é uma unidade no resultado final de uma versão impressa ou colocada na Internet. Por exemplo, quando se formata um parágrafo com um determinado tipo de letra, ou quando se coloca uma imagem num determinado ponto do écran, não estamos a tratar individualmente nem os caracteres do parágrafo nem os pixeis da imagem.

Um aspecto interessante destes objectos é que podem ser agrupados numa hierarquia de objectos gráficos: uma caixa (“box”) pode conter um ou mais parágrafos (“paragraph”). Este aninhamento de objectos começa a parecer familiar, parece mesmo a estrutura dum documento XML. Parece mas não é; um objecto gráfico é a representação pictórica dum componente lógico do documento.

Basicamente, uma folha de estilo XSL é composta por um conjunto de regras de construção. Cada regra tem duas partes, um padrão e um conjunto de acções. Sempre que um elemento do documento original XML corresponder ao padrão especificado numa regra, o correspondente bloco de acções é executado.

Exemplo: Esqueleto de uma folha de estilo XSL

```
<xsl>
  <rule>
    [regra de construção 1]
    [regra de construção 2]
    …
  </rule>
  <rule>
    [regra de construção 3]
    …
  </rule>
</xsl>
```

Voltamos ao exemplo do curso e das aulas para demonstrar o que é uma regra de construção.

Exemplo: Regra de construção

```
<rule>
  <target-element type=”Aula”/>
  <DIV font-size=”12pt” font-family=”sans-serif”
       font-style=”italic”>
    <children/>
  </DIV>
</rule>
```

As folhas de estilo em XSL podem ser muito simples ou muito complexas. Na parte expositiva deste curso serão demonstradas algumas utilizações de folhas de estilo. Para

mais informações o leitor encontrará no último capítulo dedicado à bibliografia as referências necessárias.

Serão as folhas de estilo imprescindíveis

A resposta é negativa. Podemos prescindir das folhas de estilo mas, em contrapartida, teremos de adoptar uma linguagem de programação adaptada ao XML que nos permita programar a transformação.
Existem várias possibilidades quanto à linguagem a adoptar, desde produtos comerciais até outros que não têm direitos comerciais.
Nos projectos em que me envolvi até à data a escolha tem recaído essencialmente sobre duas ferramentas: o Omnimark e o Perl; no caso do Perl utilizaram-se os módulos SGMLS.pl e SGMLS.pm, e mais recentemente o XML::DT.

A programação é feita de acordo com um modelo específico muito simples de explicar: com o elemento X vamos fazer Y.
Por exemplo, o documento XML com a informação do Curso poderia ser transformado em HTML com a seguinte script Omnimark.

```
Exemplo: script Omnimark
  ELEMENT CURSO
    OUTPUT "<HTML>%c</HTML>"
  ELEMENT TITULO
    OUTPUT "<H1>%c</H1>
  ELEMENT AUTOR
    OUTPUT "<I>%c</I>
  ELEMENT RESUMO
    OUTPUT "<DIV>%c</DIV>
  ELEMENT PARA
    OUTPUT "<P>%c</P>"
```

Na parte expositiva do curso serão mostrados várias exemplos nas várias linguagens e serão propostos alguns exercícios.

Extensible Linking Language (XLL) e Extended Pointers (XPointer)

Uma das características que tornou o HTML popular foi a possibilidade de inserir hiperligações numa página que a ligassem a outras páginas. Foi assim que o foi criado o grande manancial de informação que é o WWW.
O XML pretende ir mais além e ultrapassar algumas das limitações do mecanismo de hiperligações do HTML. O XLL tem duas partes: o XLink dedicado à especificação de ligações e o XPointer dedicado ao endereçamento.

O Xlink difere do HTML porque vai tentar ultrapassar as limitações existentes, nomeadamente:

- Ligações multi-direccionais – a possibilidade de uma ligação poder ser atravessada em ambos os sentidos.
- Ligações com destinos múltiplos – dar a possibilidade ao utilizador de escolher um de entre vários destinos.
- Pode operar com um repositório de endereços e toda a gestão a este associada.
- Permite a colocação de ligações "out-of-line" – podemos ter um ficheiro à parte onde será colocada a informação dos ligações (quais são as suas extremidades); este ficheiro acaba por ser uma lista de pares; este mecanismo é um passo na implementação das ligações bi-direccionais.

O Xpointer vem complementar o Xlink. Normalmente quando temos uma ligação duma página a outra, e se indica ao browser que se quer seguir essa ligação, o browser carrega a nova página integralmente. A ideia por detrás do Xpointer é optimizar esta funcionalidade. Para isso, disponibiliza uma pequena linguagem de query que permite seleccionar qual a parte da nova página que se quer ver. Apresenta-se a seguir um exemplo.

```
Exemplo: XPointer
  http://www.di.uminho.pt/~jcr/CURSOS/curso.xml#
         ROOT()CHILD(4,Aula)
```

Este endereço apontaria para a quarta aula no documento estruturado XML de nome curso.xml

O Xlink e o Xpointer são bastante complexos e mereceriam um capítulo inteiramente dedicado a eles mas o facto de não serem ainda suportados por nenhuma ferramenta faz deles apenas um objecto de estudo a ter em conta. No entanto, quem quiser ler mais sobre o assunto pode consultar o livro "the XML Companion".

Conclusão

Neste documento, fez-se uma travessia horizontal do standard XML.
Certamente, muita coisa ficou por dizer.
A ideia será a de tentar aprofundar os conceitos e as metodologias pelas quais as pessoas tenham mais interesse. O curso será dado numa perspectiva teórico-prática, com muitos exercícios e aplicações, onde a audiência será convidada a intervir e a prestar o seu contributo.

Bibliografia

[1] **Extensible Markup Language (XML) Version 1.0**
World Wide Web Consortium Recommendation 10-February-1998
http://www.w3.org/TR/1998/REC-xml-19980210.html

[2] **Cascading Style Sheets Level 1 (CSS1)**
World Wide Web Consortium Recommendation 17-December-1996
http://www.w3.org/TR/REC-CSS1-961217

[3] **Cascading Style Sheets Level 2 (CSS2)**
World Wide Web Consortium Recommendation 12-May-1998
http://www.w3.org/TR/1998/REC-CSS2-19980512/

[4] **Extensible Stylesheet Language (XSL) Version 1.0**
World Wide Web Consortium Working Draft 18-August-1998
http://www.w3.org/TR/1998/WD-xsl-19980818

[5] **Namespaces in XML (XML Namespace**)
World Wide Web Consortium Recommendation 14-January-1999
http://www.w3.org/TR/1999/REC-xml-names-19990114/

[6] **Extensible Linking Language (XLink)**
World Wide Web Consortium Working Draft 3-March-1998:
http://www.w3.org/TR/1998/WD-xlink-19980303

[7] **XML Pointer Language (XPointer)**
World Wide Web Consortium Working Draft 3-March-1998
http://www.w3.org/TR/1998/WD-xptr-19980303

[8] **Level 1 Document Object Model Specification Version 1.0**
World Wide Web Consortium Working Draft 20-July-1998
http://www.w3.org/TR/1998/WD-DOM-19980720/

[9] **Schema for Object-oriented XML (SOX)**
World Wide Web Consortium Note 15-September-1998
http://www.w3.org/TR/1998/NOTE-SOX-19980930/

[10] **XML-QL: A query language for XML**
World Wide Web Consortium Note 19-August-1998
http://www.w3.org/TR/1998/NOTE-xml-ql-19980819/

[11] **Web Interface Definition Language (WIDL)**
World Wide Web Consortium Note 22-September-1997
http://www.w3.org/TR/NOTE-widl-970922

[12] **Vector Markup Language (VML)**
World Wide Web Consortium Note 13-May-1998
http://www.w3.org/TR/1998/NOTE-VML-19980513

[13] **HyperText Markup Language Version 4.0 (HTML)**
World Wide Web Consortium Recommendation 24-Apr-1998
http://www.w3.org/TR/1998/REC-html40-19980424/

[14] **Document Content Description for XML (DCD)**
World Wide Web Consortium Note 31-July-1998
http://www.w3.org/TR/1998/NOTE-dcd-19980731.html

[15] **Just XML**
John E. Simpson
ISBN 0-13-943417-8
http://www.phptr.com

[16] **XML Complete**
Steven Holzner
ISBN 0-07-913702-4

[17] **Document Style Semantics and Specification Language (DSSSL)**
ISO/IEC standard 10179:1996
ftp://ftp.ornl.gov/pub/sgml/wg8/dsssl/dsssl96b.ps.Z

[18] **XSL Tutorial**
Microsoft
http://www.microsoft.com/workshop/c-frame.htm#/workshop/xml/xsl/tutorial/functions.asp

[19] **Mathematical Markup Language (MathML) 1.0 Specification**
World Wide Web Consortium Recommendation 7-April-1998
http://www.w3.org/TR/1998/REC-MathML-19980407/

[20] **SAX 1.0: The Simple API for XML**
http://www.megginson.com/SAX/

[21] **The SGML Handbook**
Charles F. Goldfarb
ISBN: 0-19-853737-1

[22] **Hypermedia/Time-based Structuring Language (HyTime)**
ISO/IEC 10744-1992(E)
http://www.ornl.gov/sgml/wg8/docs/n1920/html/n1920.html

[23] **Guidelines for Electronic Text Encoding and Interchange**
C. M. Sperberg-McQueen and *Lou Burnard*
Association for Computers and the Humanities (ACH), Association for Computational Linguistics (ACL), and Association for Literary and Linguistic Computing (ALLC).

[24] **Uniform Resource Locators**
Internet Engineering Task Force (IETF). RFC 1738 1991.
http://www.w3.org/Addressing/rfc1738.txt

[25] ***Uniform Resource Identifiers (URI): Generic Syntax and Semantics***
Berners-Lee, T., R. Fielding, and *L. Masinter*. 1997
(Work in progress; see updates to RFC1738.)

[26] **XML-Data**
World Wide Web Consortium Note 05-Jan-1998
http://www.w3.org/TR/1998/NOTE-XML-data-0105/

[27] **ECMAScript Language Specification**
Standard ECMA-262 2nd edition. June 1998
http://www.ecma.ch/stand/ecma-262.htm

[28] **XML Parser for Java**
IBM AlphaWorks
http://www.alphaworks.ibm.com/formula/XML

[29] **the XML Companion**
Neil Bradley – Addison Wesley
ISBN 0-201-34285-5